Implementing Boolean Functions in Hybrid Digital-Analog Systems

Vivek Kohar,[1,*] Behnam Kia,[1] John F. Lindner,[2] and William L. Ditto[1]
[1]Department of Physics, North Carolina State University, Raleigh, North Carolina 27695-8202, USA
[2]Physics Department, The College of Wooster, Wooster, Ohio 44691, USA
(Received 7 November 2016; revised manuscript received 27 February 2017; published 11 April 2017)

We propose an architecture to implement multi-input one-output Boolean functions using chaos computing in hybrid digital-analog systems consisting of a digital block of conventional AND gates and a nonlinear circuit. This architecture efficiently utilizes the superstable initial conditions of a nonlinear circuit and enables us to implement all possible 2^{2^m} Boolean functions of m data inputs in just m iterations of the nonlinear circuit, resulting in better operating speed and noise tolerance. In an ideal nonlinear map, this architecture eliminates the need for a decoder, as the outputs are mapped to maxima and minima of the map and can be fed directly to the next stage, enabling multilayer concatenation. We demonstrate the utility of this architecture in a three-transistor circuit.

DOI: 10.1103/PhysRevApplied.7.044006

I. INTRODUCTION

The impending failure of Moore's law has led to the development of novel approaches such as nonlinear or chaos computing (CC) [1–4] to get more computation out of a limited number of transistors. It was shown recently that a simple nonlinear circuit can theoretically implement an infinite number of functions [5], which in practice are limited by the amplification of ambient noise with each iteration. Researchers have described an integrated circuit implementation of CC with a nonlinear circuit [6] as well as other transistor-based implementations of chaotic circuits [7,8]. CC allows us to get different multi-input one-output functions using the same hardware giving us a reconfigurable logic block. We can change the functionality of the logic block by changing the control signals and get any function that we need. Moreover, this hardware is fully compatible with conventional digital systems and can be fabricated on integrated circuits using the same semiconductor fabrication technologies.

Manufacturing nonidealities and ambient noise limit the number of functions that we can obtain from these nonlinear circuits in practice. The ambient noise is amplified exponentially when the circuits are operating in the chaotic regime, thereby limiting the number of iterations for which we can use the nonlinear circuit. The number of implementable functions increases exponentially with the number of iterations [5], so it is crucial that we can maximize the iterations for which output can be obtained successfully. One method to achieve this objective is to suppress the noise evolution through coupled redundant nonlinear circuits as the local noise in different circuits averages out and the overall deviation in the output is reduced [9–11].

In this article, we propose an alternate method which exponentially decreases the number of iterations required to implement all Boolean functions. This method utilizes a hybrid digital-analog system consisting of conventional digital circuits and an analog nonlinear circuit and distributes the computation between the digital and nonlinear circuit. The increase in hardware complexity due to digital block is compensated to a large extent by a corresponding reduction in the complexity of the encoder. Hybrid digital-analog systems have been proposed as emerging technologies for transmission [12,13], wireless video coding [14], control systems [15], and chaotic circuits [16].

II. THEORETICAL FRAMEWORK

Let us assume that both data, as well as control signals, are Boolean variables and we want to implement all Boolean functions f: $B^m \rightarrow B$, where $B = \{0, 1\}$ of m data inputs. The m Boolean data inputs, $d_1, d_2, ..., d_m$ result in 2^m distinct input combinations as each input can take two values, namely, 0 and 1. The output corresponding to these input combinations can be 0 or 1 and, consequently, the number of all possible multi-input one-output Boolean functions for m data inputs is 2^{2^m}. For example, two data inputs d_1 and d_2 result in 2^2 different input combinations 00, 01, 10, and 11 and $2^{2^2} = 16$ distinct functions corresponding to an output of 0 or 1 for these four input combinations. For a CC element to yield all possible Boolean functions, we need 16 different control signals to select a specific function to be implemented. These control signals can be provided by four control inputs c_1, c_2, c_3, and c_4. In other words, we need control signals $c_1, c_2, \ldots, c_{2^m}$ to implement all m-input one-output functions.

CC is a paradigm to utilize the nonlinearity of physical systems to obtain Boolean functions [4,17]. The primary advantage of CC elements is their reconfigurability, which

*vkohar@ncsu.edu

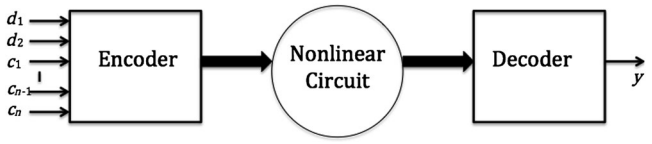

FIG. 1. Schematic diagram of conventional nonlinear or chaos computing.

allows the extraction of different Boolean functions from the same hardware. Control signals are used to select a specific Boolean function to be implemented. In the conventional CC approach, an encoder takes the digital data inputs and control signals as input and generates an analog output that is transformed to an initial condition of a nonlinear circuit. The nonlinear circuit initialized with this value is iterated for a fixed number of iterations and the output is then fed to a decoder which transforms it to a digital output, as shown in Fig. 1.

One can employ different encoding schemes to convert data inputs and control signals to an analog value. A simple example is the digital-to-analog converter (DAC), for which the encoding can be expressed as

$$x_e = \frac{2^n d_1 + \cdots + 2^{n-m-1} d_m + 2^{n-m} c_1 + \cdots + 2^0 c_{n-m}}{2^{n+1}}, \tag{1}$$

where n is equal to the sum of the number of data inputs and control signals. Every possible combination of data inputs and control signals is mapped to a distinct analog value yielding a total of distinct 2^n analog values. The nonlinear circuit is initialized with an initial condition x_0 given by

$$x_0 = g(x_e), \tag{2}$$

where $g(x)$ is typically a scaling function and depends on the characteristics of the nonlinear circuit. For example, if the nonlinear circuit consists of transistors with input voltages in [0 V, 5 V], then $x_0 = g(x_e) = 5x_e$. Similarly, the decoder is typically a thresholding circuit such that the output y of the decoder can be given as

$$y = D[x_T] = \begin{cases} 0 & x_T \le \tau \\ 1 & x_T > \tau \end{cases}, \tag{3}$$

where τ is threshold and T is the number of iterations of the nonlinear circuit.

The number of functions that can be obtained depends on the computing exponent [5] of the nonlinear circuit, and the maximum number of functions can be obtained in the chaotic region. The actual number of functions that we obtain is less than 2^{2^m}, as some of the control functions yield the same function even in the chaotic region [5] and all possible functions can be obtained only if $n \gg 2^m$. Moreover, a large number of iterations are needed to obtain these functions. For example, for four input bits and 16 control signals, we get about 16 distinct functions after six iterations and 56 functions after ten iterations [5]. This number saturates at about 22 000 after 30 iterations, which are about one-third of all possible functions (65 536). Ambient noise and finite resolution of experimental devices severely restricts the maximum number of iterations. For example, the reliability of a recently fabricated nonlinear circuit [6] starts decreasing from the sixth iteration onwards. Thus, it is imperative to devise an approach that can fully utilize the potential of the nonlinear circuit and yield maximum functions in a minimum number of iterations.

We propose the hybrid architecture shown in Fig. 2. It optimally utilizes the potential of chaos computing, as it can theoretically implement all Boolean functions with a minimum number of control signals. In this architecture, the data inputs and control signals are first passed through a digital block to generate all possible combinations of data inputs, and then each combination is associated with a control signal (see Appendix). The number of combinations of data inputs can be found using the binomial coefficients C_k^m, which denote the possible number of combinations of m data inputs considered k at a time. From the properties of binomial coefficients we know that $\sum_{k=0}^{m} C_k^m = 2^m$, and hence the number of control signals that we need agrees with the number we had found previously.

The underlying principle behind this architecture is that various Boolean functions can be expressed as logical functions of combinations of data inputs and the control signals determine whether or not a specific combination contributes to the output of a function. The analog circuit including the nonlinear circuit determines whether the selected combinations should be mapped to 0 or 1. In this architecture, the encoder is an averaging circuit which is much simpler than a DAC as needed in the conventional architecture and compensates for the increased hardware necessity of the digital block. The encoded analog value x_e of the encoder can be given as

$$x_e = \frac{\sum_{i=1}^{2^m} c_i \wedge X_i}{2^m}. \tag{4}$$

For example, for two data inputs, we have

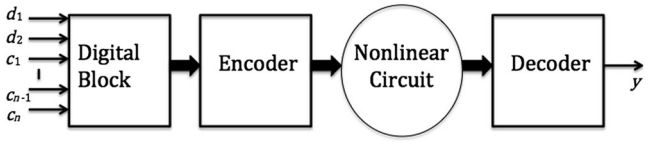

FIG. 2. Schematic diagram of chaos computing in hybrid digital-analog systems.

$$x_e = \frac{c_1 + c_2 \wedge d_1 + c_3 \wedge d_2 + c_4 \wedge d_1 \wedge d_2}{2^2}. \quad (5)$$

Note that the encoder in this scheme produces only $2^m + 1$ encoded values as opposed to 2^{2^m} encoded values needed in Eq. (1). In other words, the number of initial conditions that are to be resolved decreases exponentially in this hybrid architecture. The number of distinct functions to which an encoded value contributes is simply the binomial coefficient corresponding to that encoded value, i.e., $C_k^{2^m}$ for $0 \le k \le 2^m$. The sum of these coefficients will be 2^{2^m}, which is equal to the maximum number of Boolean functions for m data inputs.

In this architecture, each encoded value x_e is mapped to a superstable initial condition of the nonlinear circuit and the functional form of $g(x)$ depends on the nonlinear circuit. For a generic map, $x_i = f(x_{i-1})$, the superstable initial conditions for iteration i are the initial conditions x_0, for which $(d/dx)f^i(x_0) = 0$ or $f^i(x_0)$ is a maxima or minima of $f^i(x)$. Using Taylor's expansion for an initial condition $(x_0 + \delta)$ representing a small perturbation δ, of the initial condition x_0, we can approximate the ith iteration for $x_0 + \delta$ by $f^i(x_0) + \delta(d/dx)[f^i(x_0)]$ after neglecting the second and higher-order terms. If $(d/dx)[f^i(x_0)] = 0$, then the second term will be zero and the difference between $f^i(x_0)$ and $f^i(x_0 + \delta)$ is proportional to higher-order terms which are very close to zero if δ is small. So, the superstable initial conditions of the nonlinear circuit are robust against perturbations [11] and map to maxima and minima of the response function of the nonlinear circuit.

If the nonlinear circuit is piecewise linear with the same domain and codomain like a tent map, then $g(x)$ is simply a scaling function as in Eq. (2). For higher-order nonlinearity, $g(x)$ is a nonlinear function as the superstable initial conditions are nonuniformly distributed. The nonlinear circuit maps the neighboring superstable initial conditions to different output values. If the domain and codomain of the response function of the nonlinear circuit are same, then, the maxima and minima correspond to 0 and 1, respectively, and a decoder is not needed. If the domain and codomain are different, then a decoder is needed, which can be the same as the conventional CC decoder given in Eq. (3).

A comparison of conventional and hybrid architectures for chaos computing is shown in Table. I. The digital block in the hybrid digital-analog architecture adds to the complexity of the circuit, but the overall complexity of this approach is lower or comparable to the conventional approach. In the conventional approach, the encoder is a $(m + 2^m)$-bit digital-to-analog converter, whereas a $(2^m + 1)$ bit averaging circuit is used as an encoder in the hybrid approach. The VLSI implementation of the hybrid CC encoder is simpler and occupies less space and compensates for the additional digital-block hardware. The elimination of the decoder further reduces the space complexity of the hybrid approach. The hybrid approach is better in terms of time complexity also as the time complexity of the nonlinear circuit is much lower and compensates for the increased time complexity due to the digital block.

TABLE I. Comparison of conventional and hybrid digital-analog chaos computing.

Feature	Conventional chaos computing	Hybrid digital-analog chaos computing
Digital block	Not needed	AND gates
Encoder	DAC	Averaging circuit
Minimum iterations	2^m	m
Decoder	Threshold detector	Not needed
Noise tolerance	Less	More

III. EXPLICIT EXAMPLE

Now we illustrate the hybrid architecture through the two-data-input example. Let us assume that the nonlinear circuit is the tent map given by

$$x_{i+1} = f_\mu(x_i) = \begin{cases} \mu x_i & \text{for } x_i < \frac{1}{2} \\ \mu(1 - x_i) & \text{for } \frac{1}{2} \le x_i, \end{cases} \quad (6)$$

which is chaotic for $\mu = 2$ and maps an initial condition $x_0 \in [0, 1]$ to $x_{i+1} \in [0, 1]$. The superstable initial conditions for the first iteration are $x_0 = 0, 0.5, 1$, which map to 0,1,0, respectively. At the next iteration, the maxima at 0.5 changes to minima and two new maxima appear at 0.25 and 0.75, as shown in Fig. 3. One can find superstable initial conditions at any iteration i by finding the roots of the equation $(d/dx)[f^i(x)] = 0$. At any iteration i, there will be 2^{i-1} copies of the tent, and the superstable initial conditions corresponding to maxima and minima will be $0, 1/2^i, 2/2^i, 3/2^i, \ldots, (2^i - 1)/2^i, 1$. There are 2^{i-1} maxima and $2^{i-1} + 1$ minima for a total of $2^i + 1$ superstable points. This gives us a one-to-one correspondence between the number of data inputs m and the iterations i required to obtain all possible functions for those data inputs.

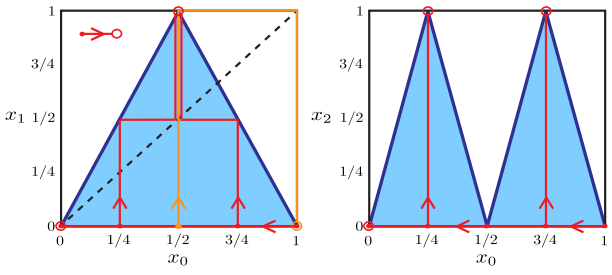

FIG. 3. Output of the tent map after one iteration (x_1) and two iterations (x_2). Mapping of superstable initial conditions x_0 to x_2 is shown by arrows.

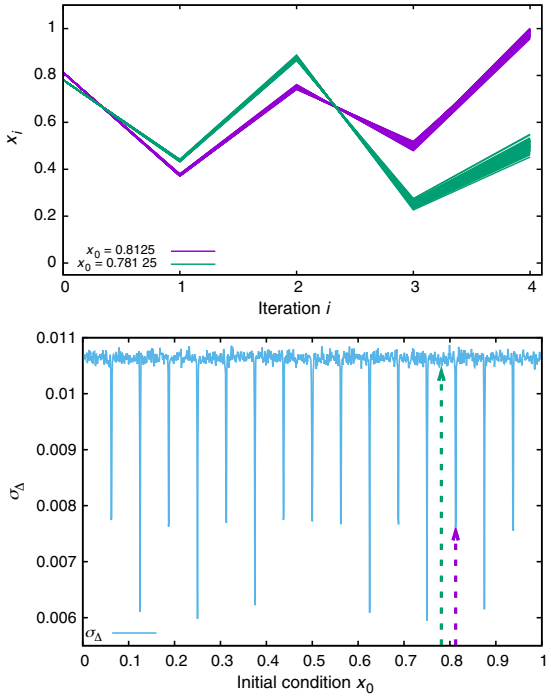

FIG. 4. Iterations of two nearby initial conditions of the tent map in the presence of ambient Gaussian noise of zero mean and standard deviation $\sigma = 10^{-3}$. One hundred trajectories are shown for each initial condition. $x_0 = 0.8125$ is a superstable initial condition for $i = 4$ and, consequently, the trajectories stay close to each other in comparison to a nonsuperstable initial condition $x_0 = 0.781\,25$. The standard deviation σ_Δ of deviation Δ of 10^4 noisy trajectories from a noiseless trajectory for initial conditions $\in [0, 1]$ after four iterations of the tent map is shown in the bottom panel. The initial conditions used in the top panel are marked by arrows.

To study the noise robustness of different initial conditions, we add noise terms to x_0 as well as at every subsequent iterate x_j for $1 \le j \le i$ such that the noisy map can be written as $x_i = f(x_{i-1} + \delta_{i-1})$, where δ_{i-1} is the instantaneous noise term with zero mean and standard deviation σ. One hundred trajectories of the tent map for two different initial conditions are shown in Fig. 4, and we observe that divergence of the superstable initial condition $x_0 = 0.8125$ is much lower compared with divergence of trajectories starting with $x_0 = 0.781\,25$. We calculate the deviation Δ as the deviation of the noisy trajectory from the noiseless trajectory and plot the standard deviation σ_Δ of these deviations in Fig. 4. We observe that the σ_Δ of superstable initial conditions is significantly less compared to σ_Δ of nearby initial conditions.

The data inputs and control signals are encoded and mapped to initial conditions according to Eqs. (5) and (2) such that $g(x) = x$. The encoded values or the corresponding initial conditions of the tent map are shared by different combinations of data inputs and control signals as sixteen possible combinations have been mapped to just five distinct initial conditions. The output of the tent map after two iterations for $x_0 \in [0, 1]$ is shown in Fig. 3. Note that we do not need a decoder in this case and the outputs corresponding to different control and data bits are given in Table II. In summary, we observe that all of the sixteen different functions for two data inputs can be implemented by using four control signals and five initial conditions of the tent map which are resolved in just two iterations.

TABLE II. Truth tables and instruction sets.

Control bits	$[d_1 d_2]$				Function
$[c_1 c_2 c_3 c_4]$	[00]	[01]	[10]	[11]	
0000	0	0	0	0	0
0001	0	0	0	1	1
0010	0	1	0	1	2
0011	0	1	0	0	3
0100	0	0	1	1	4
0101	0	0	1	0	5
0110	0	1	1	0	6
0111	0	1	1	1	7
1000	1	1	1	1	8
1001	1	1	1	0	9
1010	1	0	1	0	10
1011	1	0	1	1	11
1100	1	1	0	0	12
1101	1	1	0	1	13
1110	1	0	0	1	14
1111	1	0	0	0	15

IV. PRACTICAL APPLICATION

In this section, we demonstrate the implementation of multi-input one-output functions using a three-transistor nonlinear circuit shown in Fig. 5(a). This nonlinear circuit is implemented in a Cadence Spectre simulator using the ON C5, 0.5-μm process models provided by the vendor. The output from the Cadence simulator is fed back as its input to generate further iterations. For practical considerations, we limit the accuracy in the feedback process to 10 μV. Further implementation details are available in Refs. [5,6]. This simple circuit exhibits different behaviors

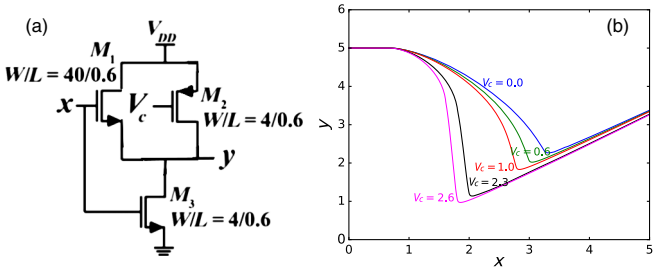

FIG. 5. (a) A three-transistor nonlinear circuit. (b) Nonlinear response curve of the circuit for different $V_c = 0$, 0.6, 1.0, 2.3, 2.6 V.

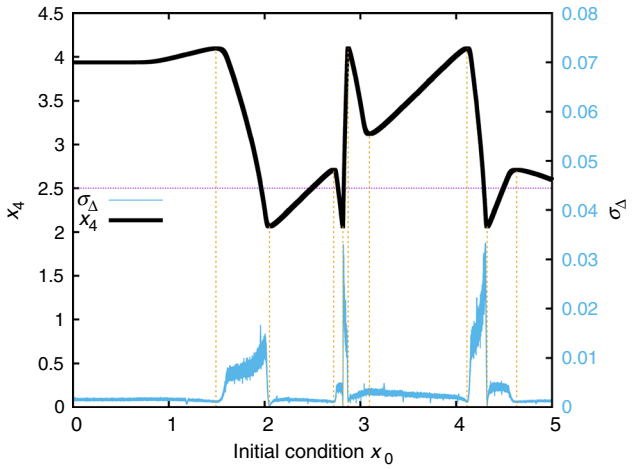

FIG. 6. Fourth iteration of the map for $V_c = 0.5$ V (thick black line) and the standard deviation σ_Δ of the deviations of a noisy map from that of a noiseless (thin blue line). The noise is assumed to be Gaussian with zero mean and 0.001 standard deviation.

depending on the bias voltage V_c which acts as a bifurcation parameter (see Fig. 8). The response y, of the circuit for various input voltages x, at different bifurcation voltages V_c, is shown in Fig. 5(b).

Conventional CC can be implemented in this circuit by using a DAC as an encoder which encodes the data inputs and control signals to an analog value according to Eq. (1) and then scaling these values to [0 V, 5 V]. If we consider the case of two data inputs and four control signals and use a six-bit DAC as the encoder, we get 64 analog values. For each control signal, there are four different analog values and each analog value corresponds to a unique combination of data inputs and control signals. Let us suppose that we iterate the map four times. The fourth iterate of the map for $V_c = 0.5$ V is shown in Fig. 6. The function implemented by a control signal can be found out by finding the output for all four analog values corresponding to that control signal. For example, the control signal 0000 results in four analog values 0, 1.25, 2.5, 3.75 V which are mapped to 3.94, 4.04, 2.51, 3.76 V or a decoded output of 1111 that corresponds to the function number 8 as per Table II. Similarly, the control signal 0001 will implement function 8. We find that even after four iterations, nearby initial conditions are not resolved by the map and we can implement few functions only [5].

To implement CC using the hybrid architecture, we pass the digital data inputs and control signals to a digital simulator and the output of the digital block is then passed through an encoder which encodes them according to Eq. (4). From Fig. 5, we observe that this nonlinear circuit maps the input values in [0 V, 5 V] to a smaller range that depends on the bias voltage V_c or, in other words, the domain and codomain for this nonlinear circuit are different. We can either limit the input range to keep the domain and codomain the same or perform a nonlinear transformation of the encoded value to map x_e to superstable initial conditions of the nonlinear circuit. We follow this second approach of nonlinear transformation and calculate the number of functions that can be implemented with m data inputs after m iterations at different bias values.

The initial conditions where the map in Fig. 6 takes a minimum or maximum value are marked by vertical lines and one can observe that the standard deviation of the deviations of the noisy map from the noiseless map is typically low for these initial conditions as compared with nearby initial conditions. Because of the asymmetry of the response of the nonlinear circuit, the number of superstable points is sometimes less than $2^m + 1$. For example, in Fig. 6 there are nine superstable initial conditions between (0 V, 5 V). Including 0 V and 5 V provides us a total of 11 initial conditions that can be used to map 11 encoded values, which is less than $2^i + 1 = 17$ for $i = 4$. In such cases, we assume that the functions which encode the data inputs and control signals to larger encoded values cannot be implemented. The number of functions corresponding to the jth encoded value x_e is equal to $C_j^{2^m}$. In the specific example, only the functions that encode all their data inputs and control signals to the first 11 values can be implemented. If we want to implement all the functions, then we need to iterate the map for more iterations so that the number of superstable initial conditions is equal to or more than $2^m + 1$.

As the domain and codomain are different, we will need a decoder in this case. We fix the decoder threshold at $\tau = 2.5$ V, so that an output from the nonlinear circuit after m iterations is taken as logic 1 only if it is greater than 2.5 V and 0 otherwise. Thus, nearby superstable initial conditions representing local minima and maxima of the nonlinear circuit are mapped to 0 and 1 by the decoder. If the maxima remains below 2.5 V or the minima is above 2.5 V, we take that as an error and the functions corresponding to such an encoded value x_e are counted as nonimplementable. For example, for the 4th iteration of the circuit for $V_c = 0.5$ V, the minima at 0 V, 3.09 V, 5 V are all above 2.5 V and result in an error. Thus, the total number of nonimplementable functions is obtained by adding the number of functions corresponding to encoded values which cannot be mapped to an initial condition due to a lesser number of superstable initial conditions as discussed in the previous paragraph and the encoded values which are mapped to initial conditions corresponding to maxima below the decoder threshold or minima above the decoder threshold.

The number of implementable functions for different numbers of data inputs is shown in Fig. 7. We observe that the number of implementable functions increases very rapidly with the number of iterations or data inputs as the map is iterated m times for m data inputs. If we iterate more than m times, then the number of implementable functions will be equal to the minimum of implementable

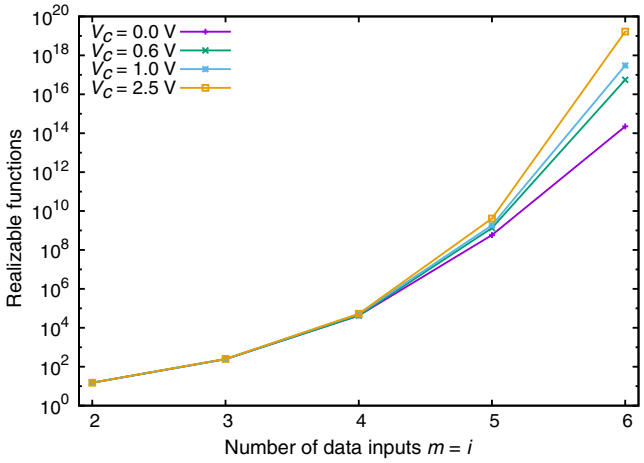

FIG. 7. Shown are the number of functions which can be implemented using the hybrid architecture. The iteration number i, after which we calculate the implementable functions is taken to be equal to the number of data inputs m. Note the exponential increase in the number of implementable functions on a log scale indicating that the increase in the number of functions is proportional to $2^{2^m} = 2^{2^i}$.

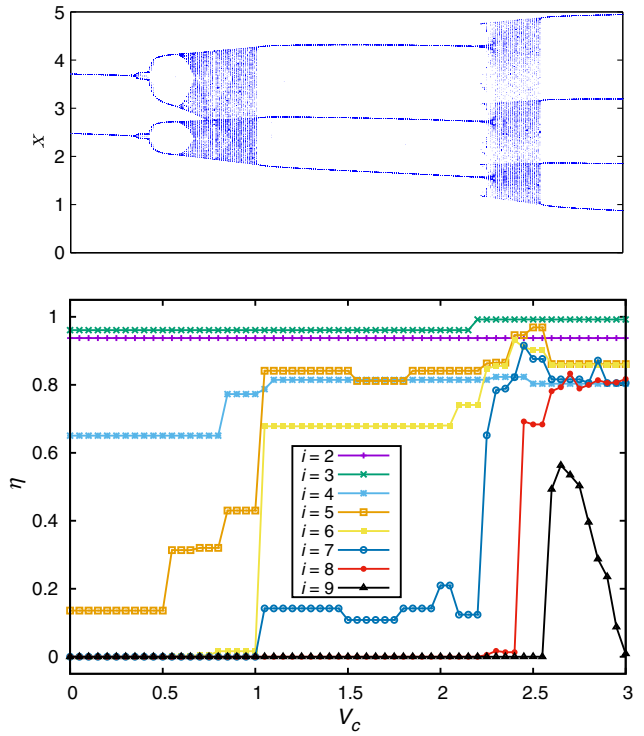

FIG. 8. The ratio (η) of implementable functions versus the maximum number of possible functions plotted for different values of bifurcation parameter V_c and the number of iterations i (data inputs m). The bifurcation diagram of the nonlinear circuit for corresponding V_c values is shown at the top.

functions for that iteration and the number of all possible functions, 2^{2^m}.

Next, we calculate the ratio η, of the number of implementable functions using this nonlinear circuit to all possible functions 2^{2^m}. We start with m data inputs, 2^m control signals, and iterate the nonlinear circuit for m iterations. The results for different m and bifurcation parameters V_c are shown in Fig. 8. We observe that the η increases near chaotic regions of the nonlinear circuit as the nonlinear circuit undergoes bifurcations. The number of implementable functions stays high in the period-three window as the basin of attraction of different fixed points is riddled. Changes in V_c also affect the maxima and minima and occasionally the number of implementable functions decreases as the maxima falls below τ or the minima is above τ. Further, the domain and codomain are different for this nonlinear circuit, and the effect of the bifurcation parameter is subdued by a decrease in the number of superstable initial conditions. These results suggest that a small digital block in our hybrid architecture can enable an exponential increase in the number of functions that can be implemented for a particular iteration number of the nonlinear circuit.

V. CONCLUSION

We have proposed an architecture to implement multi-input one-output Boolean functions in a hybrid digital-analog circuit. The digital block consists of AND gates to generate 2^m combinations of m data inputs and associate each combination with a control signal. The output of the digital block is used as an input to an encoder which maps it to $2^m + 1$ analog values which are used to initialize the nonlinear circuit with corresponding superstable initial conditions. The nonlinear circuit resolves the initial conditions such that alternate initial conditions are mapped to maxima and minima of the map after m iterations. The output is then decoded using a simple threshold-detecting decoder. This architecture is robust to noise, as the effect of noise is at a minimum at superstable initial conditions and can scale superlinearly by coupling redundant circuits [11]. Moreover, if the domain and codomain of the nonlinear circuit are the same, then the output can be used directly as an input to the next stage, allowing an easy concatenation of CC elements. We have also demonstrated the utility of this architecture in a three-transistor nonlinear circuit. Transistor-based nonlinear circuits and digital blocks can be built on the same integrated circuit [6], and this hybrid architecture makes it possible to implement a large number of functions, as the output can be obtained in a few iterations of the nonlinear circuit and the smearing of the output due to the exponential amplification of noise can be avoided.

ACKNOWLEDGMENTS

We gratefully acknowledge support from the North Carolina State University (NCSU) under Grant

No. 201584-70677 and the Office of Naval Research under Grant No. N00014-16-1-3056 and STTR Grant No. N00014-14-C-0033.

APPENDIX: DIGITAL BLOCK DETAILS

Here, the combinations imply the logical AND ($\wedge$) of the k data inputs. For example, we have C_0^m or one combination when no input data are selected, C_1^m or m combinations when a single data input is selected, C_2^m combinations when two data inputs are selected ($d_i \wedge d_j$ for $1 \le i < m$ and $i < j \le m$) and C_m^m or one combination ($d_1 \wedge d_2 \wedge \ldots d_m$) consisting of all the data inputs. We need $C_2^m + C_3^m + \cdots + C_m^m$ two-input AND gates to generate these combinations as we can use the output of the $k-1$ data-input combination and generate a k-input data combination by its logical AND with another data input. Let us denote these combinations of data inputs by X_i for $1 \le i \le 2^m$. Each of these combinations X_i is then associated with a control signal c_i through an additional logical AND operation. The association with control signals will require a further $(2^m - 1)$ two-input AND gates, as no gate is required for the first control signal. Thus, the digital block requires about $(2^m - C_0^m - C_1^m) + (2^m - 1) = (2^{m+1} - m - 2)$ two-input AND gates.

[1] Julien Borghetti, Zhiyong Li, Joseph Straznicky, Xuema Li, Douglas A. A. Ohlberg, Wei Wu, Duncan R Stewart, and R. Stanley Williams, A hybrid nanomemristor/transistor logic circuit capable of self-programming, Proc. Natl. Acad. Sci. U.S.A. **106**, 1699 (2009).

[2] Fangyue Chen, Guolong He, and Guanrong Chen, Realization of Boolean functions via CNN: Mathematical theory, LSBF and template design, IEEE Trans. Circuits Syst.Video Technol. **53**, 2203 (2006).

[3] Fangyue Chen, Guanrong Chen, Qinbin He, Guolong He, and Xiubin Xu, Universal perceptron and DNA-like learning algorithm for binary neural networks: Non-LSBF implementation, IEEE Trans. Neural Networks **20**, 1293 (2009).

[4] Sudeshna Sinha and William L. Ditto, Dynamics Based Computation, Phys. Rev. Lett. **81**, 2156 (1998).

[5] B. Kia, J. F. Lindner, and W. L. Ditto, A simple nonlinear circuit contains an infinite number of functions, IEEE Trans. Circuits Syst.Video Technol. **63**, 944 (2016).

[6] B. Kia, K. Mobley, and W. L. Ditto, An integrated circuit design for a dynamics-based reconfigurable logic block, IEEE Trans. Circuits Syst.Video Technol. **PP**, 1 (2016).

[7] Lars Keuninckx, Guy Van der Sande, and Jan Danckaert, Simple two-transistor single-supply resistor–capacitor chaotic oscillator, IEEE Trans. Circuits Syst.Video Technol. **62**, 891 (2015).

[8] Chunbiao Li, Julien Clinton Sprott, Wesley Thio, and Huanqiang Zhu, A new piecewise linear hyperchaotic circuit, IEEE Trans. Circuits Syst.Video Technol. **61**, 977 (2014).

[9] Vivek Kohar, Behnam Kia, John F Lindner, and William L Ditto, Reduction of additive colored noise using coupled dynamics, Int. J. Bifurcation Chaos Appl. Sci. Eng. **26**, 1650005 (2016).

[10] Vivek Kohar, Sarvenaz Kia, Behnam Kia, John F Lindner, and William L Ditto, Role of network topology in noise reduction using coupled dynamics, Nonlinear Dyn. **84**, 1805 (2016).

[11] Vivek Kohar, Behnam Kia, John F Lindner, and William L Ditto, Superlinearly scalable noise robustness of redundant coupled dynamical systems, Phys. Rev. E **93**, 032213 (2016).

[12] Matthias Rüngeler, Johannes Bunte, and Peter Vary, Design and evaluation of hybrid digital-analog transmission outperforming purely digital concepts, IEEE Trans. Commun. **62**, 3983 (2014).

[13] Vinod M Prabhakaran, Rohit Puri, and Kannan Ramchandran, Hybrid digital-analog codes for source-channel broadcast of gaussian sources over Gaussian channels, IEEE Trans. Inf. Theory **57**, 4573 (2011).

[14] Lei Yu, Houqiang Li, and Weiping Li, Wireless scalable video coding using a hybrid digital-analog scheme, IEEE Trans. Circuits Syst.Video Technol. **24**, 331 (2014).

[15] Rafal Goebel, Ricardo G Sanfelice, and Andrew R Teel, Hybrid dynamical systems, IEEE Control Syst. Mag. **29**, 28 (2009).

[16] Jonathan N. Blakely, Roy M. Cooper, and Ned J. Corron, Regularly timed events amid chaos, Phys. Rev. E **92**, 052904 (2015).

[17] William L Ditto and Sudeshna Sinha, Exploiting chaos for applications, Chaos **25**, 097615 (2015).